COLAS Guillaume

2ème année de DUT Informatique
Institut Universitaire de Technologie de Limoges (87)
Année 2003-2004

Compte rendu de stage de fin de DUT

Sujet :

Conception et programmation de l'Interface Homme – Machine (IHM) et des communications internes et externes de l'Instrument MISOLFA (mesure de la turbulence atmosphérique de jour)

Date : du 29 mars 2004 au 5 juin 2004

Lieu : Observatoire de la Côte d'Azur
Station de Calern
2130, Route de l'Observatoire
06460 CAUSSOLS

Maître de stage : Mr Frédéric MORAND

Tuteur à l'IUT : Mme Françoise PEDEBIDAU

Observatoire de la Côte d'Azur

Sommaire

Remerciements

Je tiens surtout à remercier mon maître de stage, Mr Frédéric MORAND, Ingénieur d'études, pour son soutien, sa disponibilité durant ma période de stage ainsi que sa grande patience envers ses stagiaires et qui a énormément contribué à ce que ce stage se passe dans les meilleures conditions.

Je remercie également Mr Christian DELMAS, responsable de l'équipe d'Astrométrie et Métrologie Solaires, de m'avoir accueilli durant cette période de stage.

Je terminerai en remerciant tous les membres du personnel du plateau de Calern qui ont joué leur rôle également pour que ce stage se passe dans de bonnes conditions et avec le sourire.

Une dernière fois, un grand merci à tous ...

Introduction

Le Rapport de stage :

Ce rapport marque la fin des deux années de formation que propose le Diplôme Universitaire de Technologie. En effet, pour l'obtention du diplôme final, chaque élève doit effectuer un stage de 10 semaines minimum dans une entreprise. Ce stage a pour but de confronter l'élève avec le milieu professionnel qu'il rencontrera très certainement au cours de sa vie. Ce stage, obligatoire, est encadré par d'une part, un professeur de l'IUT qui jouera le rôle de lien avec l'élève et l'établissement de formation, ainsi que d'un maître de stage, professionnel dans l'entreprise, qui suivra le bon déroulement de l'exécution du travail demandé.

Ce stage va avant tout permettre de mettre en avant les connaissances acquises durant les deux années de formation mais aussi de plonger l'élève dans un milieu professionnel avec la mise en place d'un projet concret. L'élève sera alors confronté aux différents problèmes liés à la vie professionnelle, lui offrant ainsi une véritable expérience, indispensable pour une formation professionnalisante telle que le DUT.

Ce rapport, jouant un véritable rôle de synthèse du travail de l'élève durant son séjour dans l'entreprise, va permettre d'évaluer sa capacité à prendre du recul par rapport à ce qu'il a accompli. L'élève pourra alors faire le bilan des connaissances acquises durant sa formation et pourra goûter aux véritables responsabilités auxquelles il peut faire face durant sa vie professionnelle (capacité d'adaptation, apprentissage permanent, confrontations d'opinions, travail en équipe, hiérarchie …).

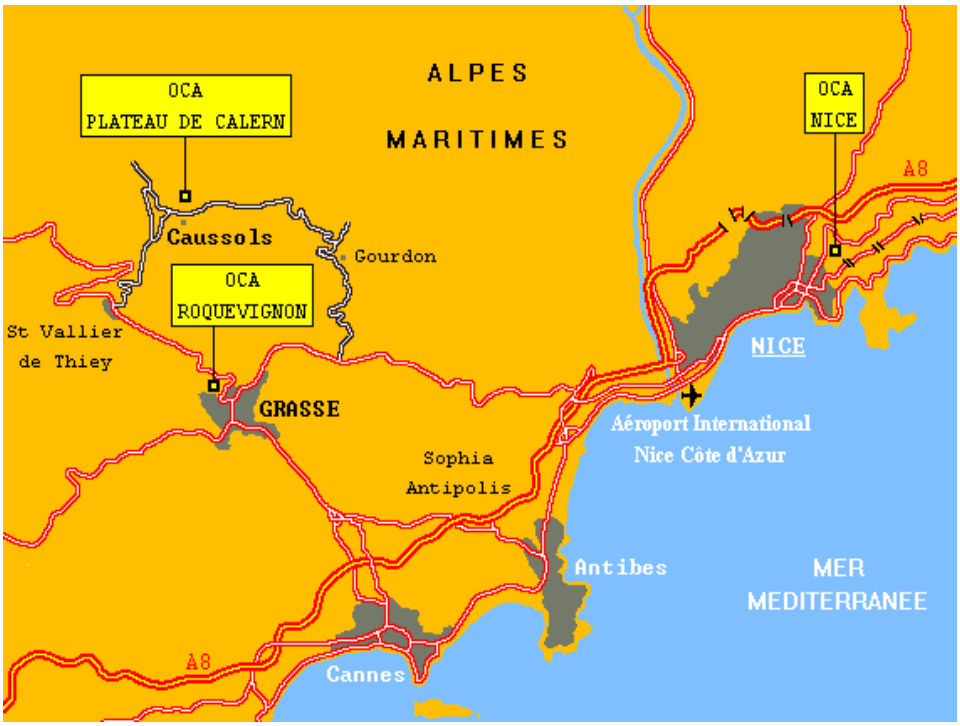
Le présent rapport va donc mettre en avant le stage que j'ai pu effectuer durant 10 semaines (du 29/03/04 au 05/06/04) à l'Observatoire de la Côte d'Azur dans le sud de la France. Il présentera, tout d'abord, l'organisme accueillant : ses rôles, ses activités ... Puis, nous verrons le sujet du stage avec un peu plus de détails. Nous passerons ensuite à une présentation des différents environnements de programmation utilisés durant le stage. Nous poursuivrons par la présentation du travail effectué avec les différentes explications et analyses concernant chaque partie. Et enfin, nous terminerons par une conclusion visant à montrer la synthèse de ces 10 semaines de stage et de vie professionnelle.

Présentation de l'Observatoire de la Côte d'Azur

L'OCA et la station de Calern :

L'Observatoire de la Côte d'Azur, établissement public à caractère administratif, dirigé par Mr. Jacques COLIN, a pour mission principale la recherche scientifique dans le domaine des sciences de l'Univers. Second observatoire national après celui de Paris, il se compose d'environ 200 personnes. Ce personnel est réparti sur 3 sites situés dans le département des Alpes-maritimes (06) :

- Le Plateau de Calern où sont réunis les instruments d'observation,
- Le Centre de Roquevignon, à Grasse,
- Le Mont Gros à Nice.



Plan des 3 sites composant l'OCA

Source : www.obs-azur.fr

Les travaux menés à l'Observatoire sont de différente nature. Les activités principales se concentrent autour :

- des observations car l'astronomie est avant tout une science d'observation ;

- des développements instrumentaux car la technologie ne cessant d'évoluer et la nécessité de voir toujours plus loin pousse à la conception de nouveaux instruments ou à l'adaptation d'anciens systèmes aux nouveaux besoins ;

- de la théorie car pour comprendre l'Univers et savoir ce qu'il faut observer, on doit tout d'abord s'appuyer sur des concepts mathématiques et des théories physiques afin de modéliser au mieux les phénomènes, les objets et les corps qui composent notre Univers;

- de la formation de nouveaux chercheurs dans le cadre de collaborations internationales.

Grâce à tout cela, les axes principaux de recherche se portent sur la détermination de la forme de la Terre, la formation et l'évolution du système solaire, la structure et les mécanismes agissant au coeur des étoiles et enfin le recensement des étoiles et des galaxies dans les domaines visibles.

Image d'un tir du Laser-Lune sur le plateau de Calern

Source : www.obs-azur.fr

Situé dans l'arrière-pays grassois, à Caussols (06), la station de Calern se situe sur un plateau à 1300 mètres d'altitude. Elle a été bâtie durant les années 70 ce qui lui permet de bénéficier d'une instrumentation relativement moderne.

(voir informations complémentaires en Annexe 1)

L'observatoire de la Côte d'Azur se compose de trois unités scientifiques et d'un département des moyens communs répartis sur les trois sites existants. Ces quatres unités sont :

- Le département Artémis dont les activités s'articulent autour des théories de la gravitation et des objets de fortes énergies, la détection interférométrique des ondes de gravitation, la métrologie laser haute résolution et l'analyse des signaux.

- Le département Cassiopée qui regroupe des chercheurs et des ingénieurs autour des thématiques comme la planétologie, la physique stellaire, l'analyse

de données en cosmologie, et l'étude des fluides et des plasmas, l'étude des turbulences en cosmologie.

- Le département Gémini qui regroupe des chercheurs et ingénieurs autour des thématiques comme l'étude des étoiles et de leur connexions dans l'environnement ainsi que la mécanique céleste et spatiale.

- Le département Galilée qui regroupe les services d'administration et de gestion de l'OCA, les services techniques et d'infrastructures : informatique, bibliothèque et ateliers.

Mon stage s'est déroulé au département Gémini, sur le plateau de Calern, plus précisément dans l'équipe d'Astrométrie et Métrologie Solaires.

Vue d'ensemble du plateau de Calern

Source : www.astrorama.net

Le département GEMINI (UMR 6203):

Créé depuis janvier 2004, suite à la restructuration des différents départements de l'OCA, le département Gémini regroupe, en partie, les anciens laboratoires CERGA (**C**entre d'**E**tudes et de **R**echerches en **G**éodynamique et **A**strométrie) et Fresnel.

Ce département s'occupe des thèmes scientifiques suivants :

- Etoiles, vents, enveloppes ;
- Instruments HRA (**H**autes **R**ésolution **A**ngulaire) et interférométrie ;
- Géodésie et mécanique céleste ;
- Télémétrie opérationnelle ;
- Métrologie-Optique et Temps-Fréquence ;
- Astrométrie solaire.

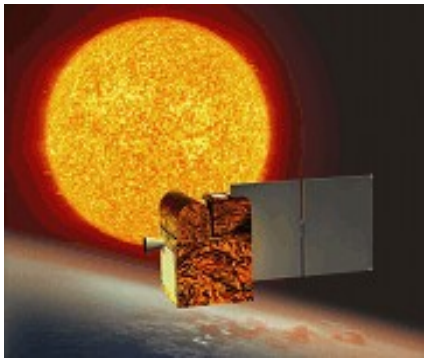
L'Astrométrie Solaire au plateau de Calern :

Francis Laclare a débuté sa série d'observations visuelles sur le Soleil en 1975 afin de mesurer les variations de son rayon au cours du temps à l'aide d'un astrolabe solaire. Depuis 1999, ces mesures sont effectuées à l'aide d'un instrument appelé DORAYSOL

(**D**éfinition et **O**bservation du **Ray**on **Sol**aire).

Après trois ans de recouvrement avec la série visuelle de F. Laclare, qui ont confirmé la qualité et la compatibilité des mesures du rayon solaire faites à Calern, il faut maintenant assurer un suivi de ce rayon dans le temps, dont l'unité pour le soleil est le cycle magnétique de onze ans (matérialisé par le nombre de taches solaires).

La mission spatiale PICARD programmée par le CNES (**C**entre **N**ational d'**E**tudes **S**patiales) pour 2008 consistera à mesurer le diamètre du soleil à quelques millisecondes d'arc près. Elle mesurera également l'irradiance solaire et la rotation différentielle de la boule de gaz sur elle-même. En comparant avec les mesures parallèles de DORAYSOL, on pourra enfin connaître avec précision l'effet de l'atmosphère terrestre sur les images et étalonner de la sorte les mesures qui devront être faites au sol pendant plusieurs cycles solaires de 11 ans. L'instrument MISOLFA aidera activement à cet objectif en mesurant les paramètres de la turbulence amosphérique.



Le satellite PICARD (vue d'artiste)

Source : www.cnes.fr

Présentation du sujet de stage

(Voir en Annexe 2 la proposition de stage envoyée)

Mon sujet présente plusieurs partie distinctes :

- La conception de l'analyseur-synthétiseur de commandes ;
- La gestion des communications distantes ;
- La création des interfaces graphiques.

La programmation de l'analyseur-synthétiseur de commandes ainsi que la gestion des communications distantes seront faites en C++ alors que la mise en place des interfaces graphiques se fera en JAVA (Le choix est encore à faire pour le centraliseur).

Le choix du langage s'est fait, pour le C++, suite aux conseils de mon maître de stage. En effet, l'outil Builder C++ de Borland possède de nombreuses propriétés pour le travail sur des chaînes nécessaires à l'élaboration de l'analyseur. Il permet, en outre, une gestion beaucoup plus simple des communications distantes et optimise les connexions par sockets qui seront utilisées dans le programme.

Pour ce qui est du JAVA, ce langage a été choisi surtout pour ses avantages de portabilité, puisque le programme devra être utilisable depuis n'importe quel poste. L'utilisateur se connectera au serveur et pourra utiliser le programme à distance. Il faut tout de même créer une interface car le système devra renvoyer des informations utiles à l'utilisateur qui concernent aussi bien l'instrument que les erreurs qui peuvent se produire durant l'utilisation du logiciel et qui, dans ce cas précis, devront être gérées par le programme lui-même.

Pour bien comprendre la manière d'aborder ce sujet, il faut tout d'abord bien comprendre le « système » général et le rôle de chaque module dans l'architecture de MISOLFA. L'apport de nombreuses documentations concernant la mise en place du projet PICARD m'a permis de voir les liens qui pouvaient exister entre les différentes machines et de comprendre les nombreux mécanismes qu'il faudrait prendre en compte. Ce travail nécessite donc une analyse rigoureuse afin de bien cerner les différentes fonctionnalitées du système. De plus, les programmes créés devront répondre à une certaine indépendance puisqu'ils devront être capables de s'adapter à n'importe quelle machine si on les transpose sur des systèmes dont les besoins sont quasi-similaires. Les modifications entre systèmes se feront par le biais de fichiers textes.

On peut voir, sur la page suivante, le synoptique du logiciel de pilotage de MISOLFA. Ce logiciel sera implanté dans le PC de pilotage et devra fonctionner comme un automate qui reçevra des consignes de l'ordinateur d'acquisition. On peut constater que le rôle de l'analyseur de commandes est très important et occupe la place centrale dans l'architecture du programme de pilotage. En effet, il va être chargé de faire le lien entre les différentes entités que regroupe ce logiciel. Toutes les informations envoyées par les différents modules transiteront par cet analyseur. Ce logiciel de pilotage, comme

nous pouvons le voir, se compose de quatre modules bien distincts :

- Un **noyau central séquenceur**, qui a pour fonction de communiquer avec les modules dits « temps réels », afin de leur transmettre des ordres de bas niveau (rotation d'une roue, commande d'un relais) ou au contraire de recevoir des paramètres (positions des axes, état d'un relais). Il doit également surveiller l'ensemble des automatismes (vérifier qu'un ordre demandé à l'électronique a été effectué correctement ; contrôler régulièrement la cohérence entre les commandes et l'état des relais, lire régulièrement les paramètres d'état de l'instrument, ...). Il doit également envoyer régulièrement aux autres parties du système (acquisition, centraliseur, affichage local) des informations sur l'état de l'instrument. Enfin il doit être capable d'effectuer des commandes de haut niveau (par exemple : pointer le Soleil), en les décomposant en séquences d'ordres à envoyer aux modules « temps réel ».

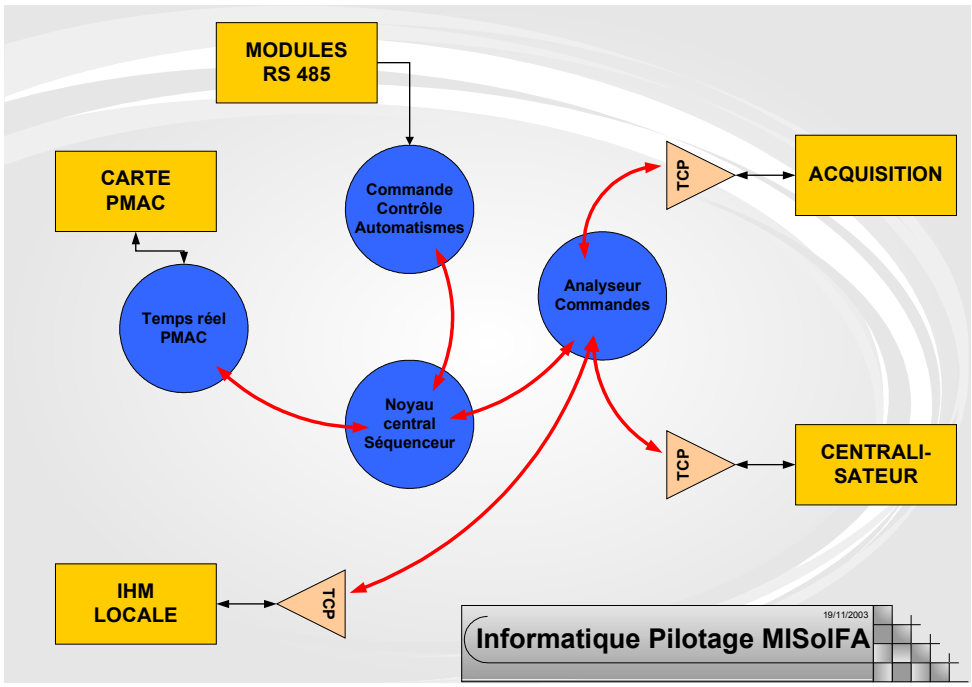
- Un **module « temps réel PMAC »** qui sert à l'interface avec la carte PMAC. Ce module, par l'intermédiaire d'une DLL, permet de traduire les commandes venant du noyau central dans le langage de la carte. D'autre part, il permet de lire les variables d'état de la carte (position codeurs, état des fins de courses, etc ...).

- Un **module « temps réel commande-contrôle automatismes »** qui envoie sur une liaison RS485 les commandes ASCII permettant de commander un relais ou de lire son état.

- Un **Analyseur-Synthétiseur de commandes** qui permet d'assurer la liaison des différentes parties du système. Cet Analyseur-Synthétiseur analyse les messages qu'il reçoit ou génère des commandes qu'il envoie. Ces messages ou commandes sont sous forme de « phrases » ASCII, consituées de mots-clés et de paramètres. Ces messages ou commandes transitent par des lignes Ethernet (protocole TCP-IP) gérées sur chaque machine par l'intermédiaire de sockets. Ces liens doivent être établis avec :

    - L'ordinateur d'acquisition,

    - Une Interface Homme-Machine locale qui permet d'accéder à toutes les informations (y compris celles de bas niveau) et à des fins de diagnostic en cas de pannes. Cette Interface doit être disponible via la console partagée par le distributeur d'entrées / sorties, *(voir annexe 3)*
    - Le centraliseur, ordinateur qui regroupe toutes les informations importantes provenant des différents instruments composant l'ensemble PICARD-SOL. *(voir chapitre suivant)*



Synoptique du logiciel de pilotage MISOLFA

Source : MISOLFA : Un moniteur de qualité d'images solaires utile pour la mission PICARD.

Le projet PICARD et MISOLFA

Le projet PICARD :

L'expérience SODISM embarquée sur le satellite PICARD (année 2008) aura pour but de mesurer avec précision le diamètre solaire. La précision obtenue dans l'espace sera bien meilleure que celle obtenue depuis les instruments au sol. Au sol, l'atmosphère terrestre « trouble » la lumière venue d'un point du Soleil sur un diamètre d'environ une seconde d'arc limitant la précision des mesures de diamètre solaire. Dans l'espace on peut espérer, avec SODISM, obtenir une précision 100 fois meilleure qu'au sol.

A cette fin, l'instrument SODISM a été conçu comme un instrument d'optique de précision. La structure du téléscope a été étudiée pour minimiser les dilatations pouvant faire varier le facteur d'échelle. De plus, l'instrument est pourvu d'un système de calibration qui mesure en permanence ce facteur d'échelle.

Le but de ce projet est de mesurer les variations de diamètre du Soleil en liaison avec le cycle solaire de 11 ans et de vérifier les études de F.Laclare, faites au Plateau de Calern. Cet instrument, mesurera les diamètres dans toutes les directions avec une précision inégalée et pourra confirmer ou infirmer de possibles irrégularités de la surface solaire. Il mesurera également les vibrations périodiques (héliosismologie).

SODISM II :

La vie du satellite étant limitée à deux ou trois ans, il sera nécessaire de continuer les mesures au sol après la fin de vie du satellite. Pour cela, un instrument identique, SODISM II sera installé au sol. Il s'agit du modèle de rechange du téléscope SODISM installé sur le satellite, que l'on va disposer sur une monture au plateau de Calern. Cet instrument, du fait de son installation au sol, aura une précision limitée par l'atmosphère terrestre.

MISOLFA :

Afin de mieux évaluer l'influence de l'atmosphère, un autre instrument dédié à la mesure des paramètres atmosphériques MISOLFA va être installé à côté de SODISM II.

MISOLFA va mesurer dès 2005 les paramètres de la turbulence atmosphérique. Pendant la vie du satellite on pourra comparer les différences entre SODISM Satellite et SODISM II avec les mesures instantanées de MISOLFA. Ensuite MISOLFA aidera à interpréter les mesures faites par SODISM II et DORAYSOL et permettra certainement d'affiner ses mesures.

(Voir en Annexe 3 l'architecture informatique et électronique de SODISM II et MISOLFA)

Image de MISOLFA et de la monture pour SODISM II

Source : Photo réalisée sur le plateau de Calern

Présentation des outils de programmation

La programmation sous Borland C++ Builder :

Le C++ est un langage de Programmation Orienté Objet (POO), dérivé du C. La POO est une méthode de structuration et de développement de programmes qui présente de profondes différences par rapport à la méthode traditionnelle de programmation procédurale. Les systèmes procéduraux nécessitent la plupart du temps une approche descendante, la structure des programmes suit le déroulement des traitements. Les systèmes orientés objet sont modelés en fonction des objets qu'ils traitent. Leur élaboration en paraît d'autant plus naturelle : au premier plan se trouvent des objets dotés de leurs caractéristiques propres, tout comme on les observe dans la réalité. Un objet est défini par une classe qui intègre des variables appelées « propriétés », et des fonctions qui agissent sur ces variables, appelées « méthodes ». Les propriétés et les méthodes peuvent être publiques ou privées, c'est à dire connues ou non des autres éléments du programme. On peut ainsi sécuriser le code, en rendant par exemple impossible, la modification d'une variable par une fonction quelconque.

C++Builder, est un environnement de développement intégré, appelé également EDI. Il contient un concepteur visuel des fenêtres Windows du programme que l'on développe, une palette de composants directement intégrable au code source, l'éditeur de code source et le débogueur. On peut ainsi passer de la représentation visuelle d'un objet, à l'éditeur de code source qui permet de changer la logique d'exécution de l'objet. Le changement des propriétés d'un objet modifie automatiquement le code source correspondant.

La programmation orientée Objet dispose d'une bonne structuration et d'une grande portabilité ce qui permettra, d'autant plus facilement, la transposition du code existant sur d'autres machines comme c'est le cas pour l'Analyseur-Synthétiseur de commandes qui doit se situer sur plusieurs machines.

La programmation sous JAVA :

Le fichier source d'un programme écrit en JAVA est un simple fichier texte dont l'extension est par convention « .JAVA ».
Ce fichier source doit être un fichier texte non formatté, c'est-à-dire un fichier texte dans sa plus simple expression, sans mise en forme particulière ou caractères spéciaux, c'est-à-dire qu'il contient uniquement les caractères ASCII de base.

Lorsque le programme est prêt à être "essayé", il s'agit de le compiler (le traduire en langage machine) à l'aide d'un compilateur. Toutefois, contrairement aux langages compilés traditionnels, pour lesquels le compilateur crée un fichier binaire directement exécutable par un processeur donné (c'est-à-dire un fichier binaire contenant des instructions spécifiques à un processeur), le code source JAVA est compilé en un langage intermédiaire (appelé pseudo-code ou bytecode) dans un fichier portant le même nom

que le fichier source à l'exception de son extension (.class).

Cette caractéristique est majeure, car c'est elle qui fait qu'un programme écrit en JAVA est portable, c'est-à-dire qu'il ne dépend pas d'une plate-forme donnée. En réalité le code intermédiaire n'est exécutable sur aucune plate-forme sans la présence d'une machine virtuelle, un interpréteur (la machine virtuelle est d'ailleurs parfois appelée interpréteur JAVA) tournant sur une plate-forme donnée, et capable d'interpréter le code intermédiaire.

Ainsi, pour peu qu'une plate-forme (windows 95 et supérieur, Unix, Linux, ...) possède une machine virtuelle fonctionnant sous son système, celle-ci est capable d'exécuter n'importe quelle application JAVA.

C'est pourquoi ce langage a été choisi pour créer les interfaces graphiques qui permettront de faire les liens entre l'instrument et un utilisateur distant. En effet, le problème de la compatibilité des systèmes distants avec l'instrument lui même sera résolu en utilisant ce type d'interface.

L'Analyseur-Synthétiseur de commandes

Description :

L'analyseur-Synthétiseur de commandes, va permet d'assurer la liaison entre les différentes parties du système composant MISOLFA. Il doit être capable d'analyser les messages qu'il reçoit ou de générer des commandes qu'il envoie. Ces messages ou commandes sont sous formes de « phrases » ASCII, constituées de mots-clés et de paramètres contenus dans des fichiers textes. On peut assimiler ce genre de programme à un système d'exploitation capable de lancer des fonctions tel LINUX en mode console ou bien MS-DOS.

La difficulté de ce travail réside principalement dans la compréhension générale de la tâche demandée puisque nous n'avons pas été forcément préparés à faire face à ce genre de travail durant nos années de formation à l'IUT. Cela demande donc une analyse rigoureuse du sujet. Les nombreuses explications de mon maître de stage m'ont beaucoup apporté tant à la compréhension du système général, qu'à la conception du programme demandé.

Une autre difficulté a été de créer un logiciel indépendant et que l'on puisse porter sur n'importe quelle machine. En effet, les seuls changements à effectuer pour passer sur un nouvel ordinateur devront s'effectuer par la modification de fichiers textes. Dans ce cas précis, nous aurons affaire à deux types de fichiers dont l'un servira au chargement de paramètres comme les répertoires de travail, le nom du fichier d'ordres, les coordonnées géographiques de l'instrument, le nombre d'images prises, ... , avec une extension « *.ini », et qui sera modifiable par le programme, et le deuxième, qui sera le fichier d'ordres, qui servira pour « apprendre » au programme les différentes commandes qu'il doit connaître avec une extension « *.txt » et qui ne doit pas être modifié.

Suite à l'initialisation du programme qui s'effectuera à chaque lancement, celui-ci devra réagir aux différents ordres qu'il reçoit, soit par le biais d'un utilisateur, soit par le biais de l'instrument lui même. Il devra donc être capable, suivant une syntaxe particulière, de prendre en compte un ordre correct avec sa suite d'arguments ou bien de le rejeter si l'ordre est « mal écrit » et générer ainsi une réaction automatique.

Il devra enfin être capable de « reconstruire » l'ordre afin qu'il soit éxécuté correctement par la machine avec la suite d'arguments demandée.

Les fichiers d'initialisation :

Le programme utilise donc deux types de fichiers dits « d'initialisation ». Le premier se compose de paragraphes dont les titres sont contenus entre crochets, puis du nom du paramètre suivi de la valeur qu'il doit prendre. Ce fichier possède l'extension « *.ini ».

(Voir un exemple de structure du fichier d'initialisation en Annexe 4)

Pour ce qui est de la gestion de ces paramètres et de la lecture de ce fichier, ce travail avait déjà été effectué auparavant lors d'un stage d'un élève en école d'ingénieur, Mr Eric HUNAUT, en 2001 avec Mr MORAND. J'ai donc repris la classe qui avait été implémentée durant ce stage puis je l'ai utilisée dans mon code. Ces classes utilisent des AnsiStrings pour lire le fichier texte. Pour récupérer un paramètre particulier, le programme cherche d'abord le paragraphe correspondant, puis le nom du paramètre et enfin récupère la valeur demandée. Les valeurs pouvant être de différentes natures, comme des « AnsiStrings », des « entiers », des « doubles » ou encore des « booléens », les fonctions de lecture de paramètres ont été surchargées.
(Pour des raisons de compréhension, un diagramme de fonctionnement du programme avec le fichier d'initialisation a été ajouté en Annexe 6)

Le deuxième fichier, quant à lui, comprend les différents ordres que l'analyseur devra connaître. En fait, il sert à tester la syntaxe de l'ordre reçu. Il faut construire ce fichier avec une syntaxe particulière pour pouvoir faire des tests sur la demande d'ordre reçu afin de vérifier si la machine est capable de construire l'ordre demandé ou non. Ce fichier possèdera l'extension « *.txt ». Le fichier d'ordres est donc construit de la manière suivante :

- « $ » en début de ligne signifie que l'on prend un nouvel ordre.
- « : » après un mot signifie que l'on vient de lire le nom de la fonction.
- « ; » après une lettre signifie que l'on vient de lire un argument.
- « ! » en bout de ligne signifie que la ligne est finie et que l'on doit lire la suivante.

Un système de « codage » a été décidé pour faciliter la lecture des paramètres dans le fichier pour chaque fonction. Une lettre remplace le type d'argument attendu :

- « i » sera utilisé pour un int (entier).
- « d » sera utilisé pour un double (float).
- « s » sera utilisé pour un String (chaine de caractères).
- « b » sera utilisé pour un bool (booléen).
- « c » sera utilisé pour un char (caractère).
- « 0 » sera utilisé lorsque la fonction d'a pas d'arguments.

Ainsi, un ordre qui s'appellera « monter » et qui attendra en paramètres un entier et un caractère, s'écrit de la manière suivante dans le fichier d'ordres :

$monter:i;c!

De même, pour un ordre s'appellant « stop » mais qui n'attend aucun paramètre, ce dernier s'écrit :

$stop:0!

(Voir un exemple de structure du fichier d'ordres en Annexe 5)

Les avantages d'une telle initialisation pour un programme sont très nombreux. En effet, de par l'utilisation dejà du langage C++, le programme sera facilement portable d'une machine à une autre. Ensuite, ce programme peut s'adapter à bon nombre de configurations, ou instruments. Par exemple, une résolution d'image particulière embarquée en tant que paramètre pourra facilement être modifiée si l'on change la caméra d'observation ; tout cela sans toucher au code source, juste en modifiant le fichier

d'initialisation associé. Cela évitera de réécrire chaque fois le programme si l'on change une partie du matériel utilisé.

L'Analyseur de commandes :

La partie d'initialisation du programme avec le fichier « *.ini » existant déjà, je ne l'ai volontairement pas prise en compte dans les détails de l'analyse suivante.

Il m'a fallu créer la partie qui permet de lire les paramètres contenus dans le fichier d'ordres. J'ai donc utilisé une analyse syntaxique, caractère par caractère, du fichier avec l'extension « *.txt » afin de pouvoir bien discerner les différentes fonctions ainsi que leurs arguments. La mise en forme syntaxique du fichier lui-même permet un contrôle plus simple. En effet, les arguments ne sont matérialisés que par une seule lettre. De plus, la délimitation de chaque mot par des caractères particuliers permet une analyse beaucoup plus simple.

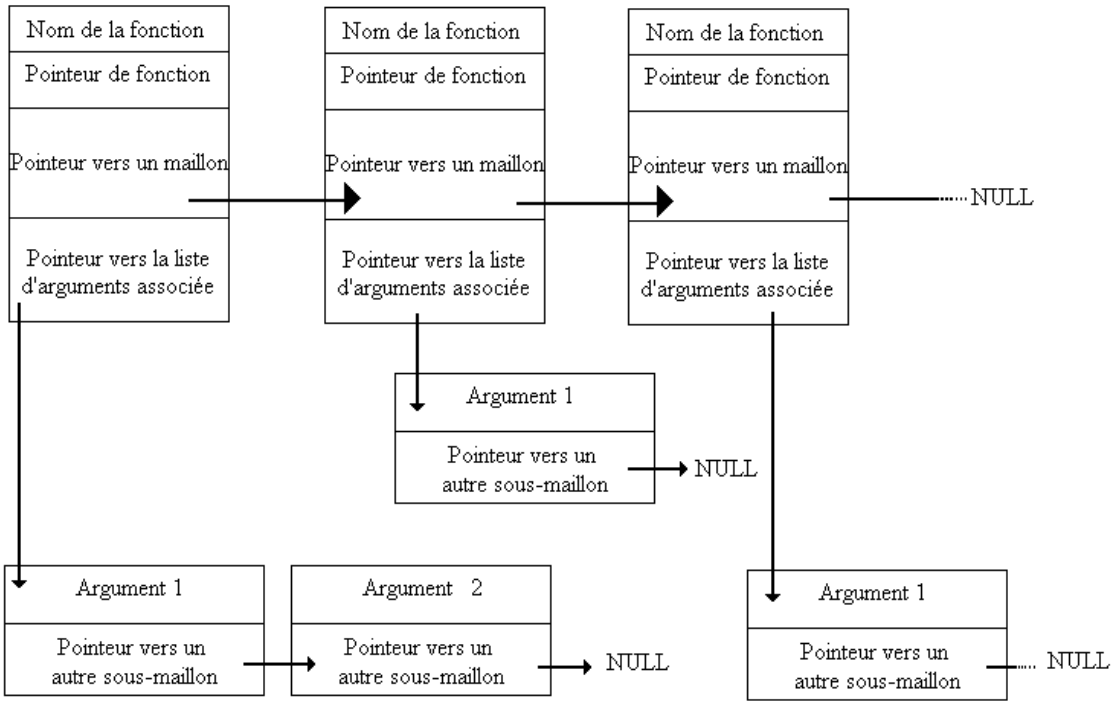
Après cela, il fallait trouver un moyen pour pouvoir garder en mémoire tout ces paramètres afin d'éviter, pour chaque réception d'ordre, un nouveau parcours du fichier. La solution que j'ai adoptée est donc l'utilisation d'une liste chaînée, qui joue le rôle de « squelette » dans mon programme. L'un des avantages essentiel de l'utilisation d'une liste chaînée est que l'on a pas à connaître par avance le nombre d'ordres dont le programme aura besoin. C'est une sorte de tableau dynamique qui permet de ne pas saturer la mémoire en prévoyant trop de place ou pas assez. Une bonne connaissance des pointeurs est primordiale car il ne fallait pas se mélanger dans la construction de cette liste chaînée. Cette dernière donc doit embarquer dans chaque maillon toutes les informations nécessaires aux futures comparaisons qui seront effectuées pour tester la validité d'un ordre ainsi que pour l'exécuter correctement.

La construction de cette liste chaînée, assez complexe, nécessite une certaine attention concernant la mise en place des différents maillons à l'aide de pointeurs. Chaque maillon correspond à un ordre précis avec sa suite d'arguments et toutes les informations nécessaires à l'éxécution de l'ordre demandé. Pour chaque ordre, il fallait également conserver les différents types d'arguments attendus. De plus, le fait de ne pas connaître le nombre d'arguments correspondant à la fonction ne permettait pas de conserver les paramètres dans un maillon. J'ai donc pour cela décidé d'utiliser une deuxième liste chaînée dans chaque maillon de la première.

A chaque réception d'un ordre, l'analyseur va donc effectuer le parcours de la liste chaînée en s'arrêtant sur chaque maillon pour analyser si l'ordre demandé correspond parfaitement aux informations contenues dans chaque maillon. Le nom de l'ordre sera d'abord comparé puis ensuite sa liste d'arguments.

L'exécution d'un ordre se fera ensuite grâce à un pointeur de fonction qui permettra de faire le lien avec la véritable fonction qui est programmée. Ce pointeur est initialisé durant la phase de lecture du fichier d'ordres. Il faudra ensuite passer simplement les paramètres adéquats à la fonction demandée.

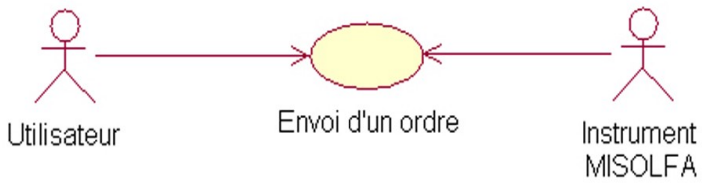
On pourra noter, ci-dessous, un exemple de construction de cette liste chaînée afin de bien comprendre sa structure.



Exemple de construction de la liste chaînée

Analyse de l'Analyseur-Synthétiseur de commandes :

1°/ Description du diagramme de cas d'utilisation :



2°/ Description du cas d'utilisation :

Description :

Ce programme totalement indépendant doit être capable de lancer des fonctions via des ordres saisis par l'utilisateur ou bien que l'instrument enverra de lui-même. Ces

ordres sont suivis des arguments nécessaires au fonctionnement de la fonction désirée.
Les noms des fonctions que l'on veut utiliser seront stockées dans un fichier texte à part d'où la nécessité de créer un programme indépendant et qui fera seul le lien avec le fichier d'ordres (celui qui contient les fonctions).

Flot Principal :

1) Lancement de l'application.

2) Le système fait l'initialisation à partir du fichier « *.ini » pour charger les différents paramètres. *(voir schéma de fonctionnement en Annexe 6)*

3) Le système fait l'initialisation à partir du fichier contenant les ordres et les paramètres.
   3.1) Le programme principal ouvre le fichier d'ordres.
   3.2) Le système lit une ligne du fichier texte.
      3.2.1) On crée un nouveau maillon.
      3.2.2) On donne à une variable AnsiString dans le maillon le nom de la fonction.
      3.2.3) On initialise le pointeur de fonction afin qu'il puisse pointer vers la fonction correspondante.
      3.2.4) On crée un sous maillon pour pouvoir enregistrer les types d'arguments pour chaque ordre.
         3.2.4.1) On lit l'argument et on enregistre son type dans un sous-maillon.
         3.2.4.2) On crée un nouveau sous-maillon.
      3.2.5) On recommence 3.2.4) jusqu'à ce que l'on arrive au caractère de fin de ligne.
      3.2.6)On fait pointer le dernier sous-maillon vers NULL.
      3.2.7) On fait pointer le pointeur du maillon actuel sur un nouveau maillon que l'on créé et qui sera considéré comme le suivant.
      3.2.8) On se positionne sur ce dernier maillon.
   3.3) On recommence 3.2) jusqu'à ce que l'on soit à la dernière ligne du fichier texte.
   3.4) On fait pointer le pointeur suivant du dernier maillon sur NULL.

4) L'Analyseur se met en attente d'un ordre.

5) L'Analyseur reçoit un ordre.

6) On parcourt la chaine pour voir si l'ordre existe.
   6.1) Pour chaque maillon on vérifie d'abord si l'ordre reçu est équivalent au nom enregistré dans l'AnsiString.
   6.2) Une fois l'ordre trouvé, on vérifie si la fonction possède ou non des arguments.
   6.3) On vérifie ensuite si le type de chaque argument est bien correct.

7) On utilise le pointeur de fonction du maillon trouvé pour appeller la fonction souhaitée.

8) On exécute la fonction.

Flot Alternatifs :

<Erreur Accès fichier Initialisation>:

Au point 2), si le système n'arrive pas à établir le lien avec le fichier d'initialisation, un message est affiché à l'écran le signalant. L'utilisateur doit alors vérifier que le fichier d'initialisation est bien présent.

1. <Erreur Accès fichier Ordres>:

Au point 3), si le système n'arrive pas à établir le lien avec le fichier d'ordres, un message est affiché à l'écran le signalant. L'utilisateur doit alors vérifier que le fichier d'ordres est bien présent.

3. <Erreur mot-clef>:

Au point 6.1), si le système termine sur un maillon qui pointe vers NULL, c'est qu'il n'a pas trouvé de fonction correspondante au mot-clef que l'analyseur a reçu. Le système affiche donc un message signalant à l'utilisateur que le mot-clef n'existe pas.

4. <Erreur nombre d'arguments>:

Au point 6.2), le système regarde le nombre d'arguments reçus et si le pointeur de sous-maillon pointe vers la valeur « 0 » (pas d'arguments) ou une autre (arguments en attente). Si le mot-clef reçu contient des arguments alors qu'il ne devrait pas y en avoir, on passe à un maillon suivant. On fait de même si le cas inverse se produit. Retour au point 6).

5. <Erreur type d'arguments>:

Au point 6.3), le système compare le type de chaque argument reçu avec la chaine de sous maillon obtenue durant l'initialisation. Si un type d'argument diffère, on passe au maillon suivant et on recommence au point 6).

6. <Erreur fin de recherche sur le nombre ou type d'argument>:

Au point 6), si le nom de fonction existe bien mais que la recherche se termine soit parce que le nombre d'arguments tapés est erroné, soit parce que le type d'argument diffère, le système affiche un message disant à l'utilisateur qu' il y a une erreur portant sur les arguments saisis (nombre ou type).

Flots d'exception :

Aucun.

Exigences particulières :

Aucune.

Pré-conditions :

Aucune.

Post-conditions :

Un ordre est généré puis exécuté.

Points d'inclusion :

Aucun.

Points d'extension :

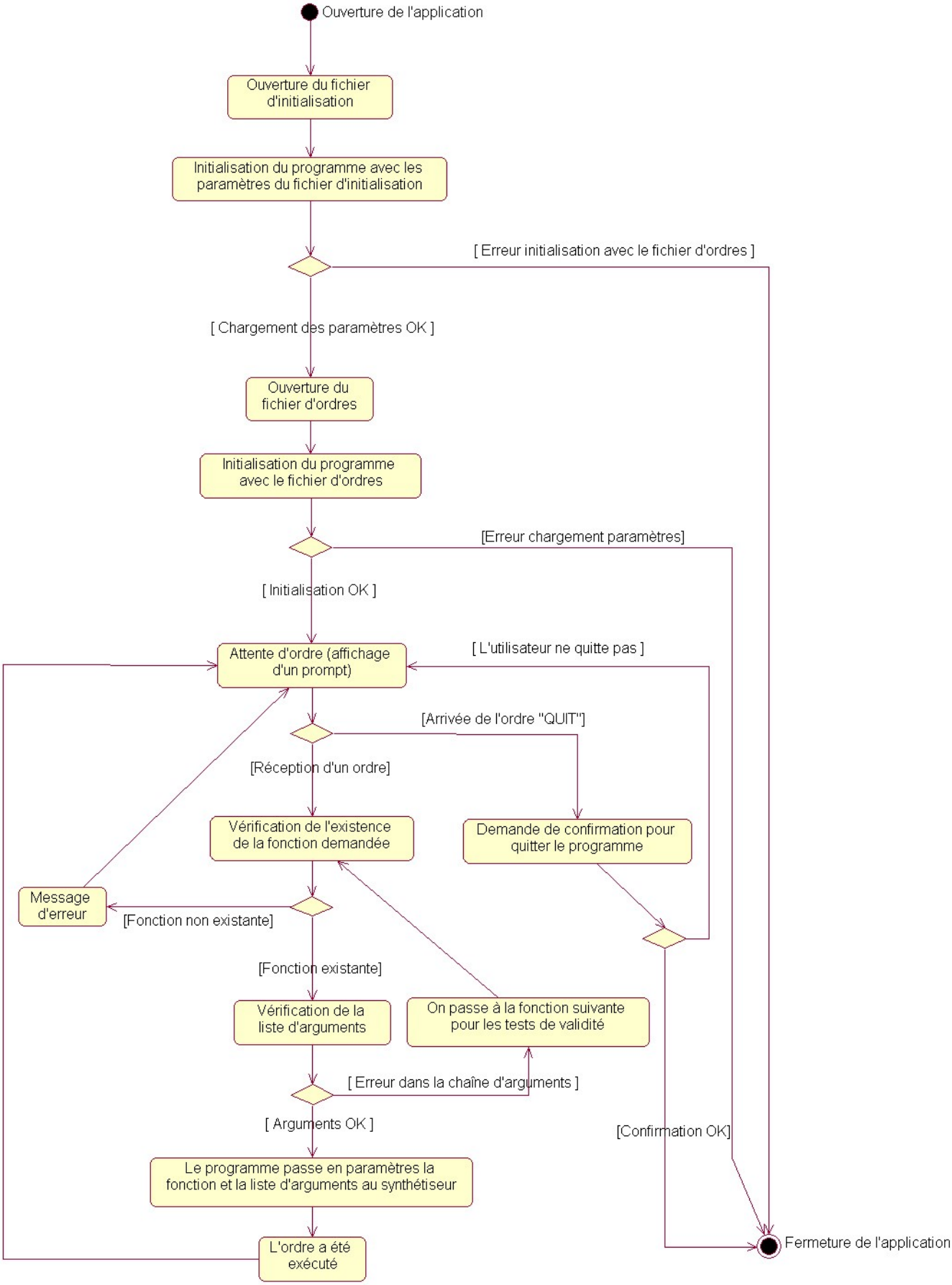
Aucun.

Liste des acteurs participants :

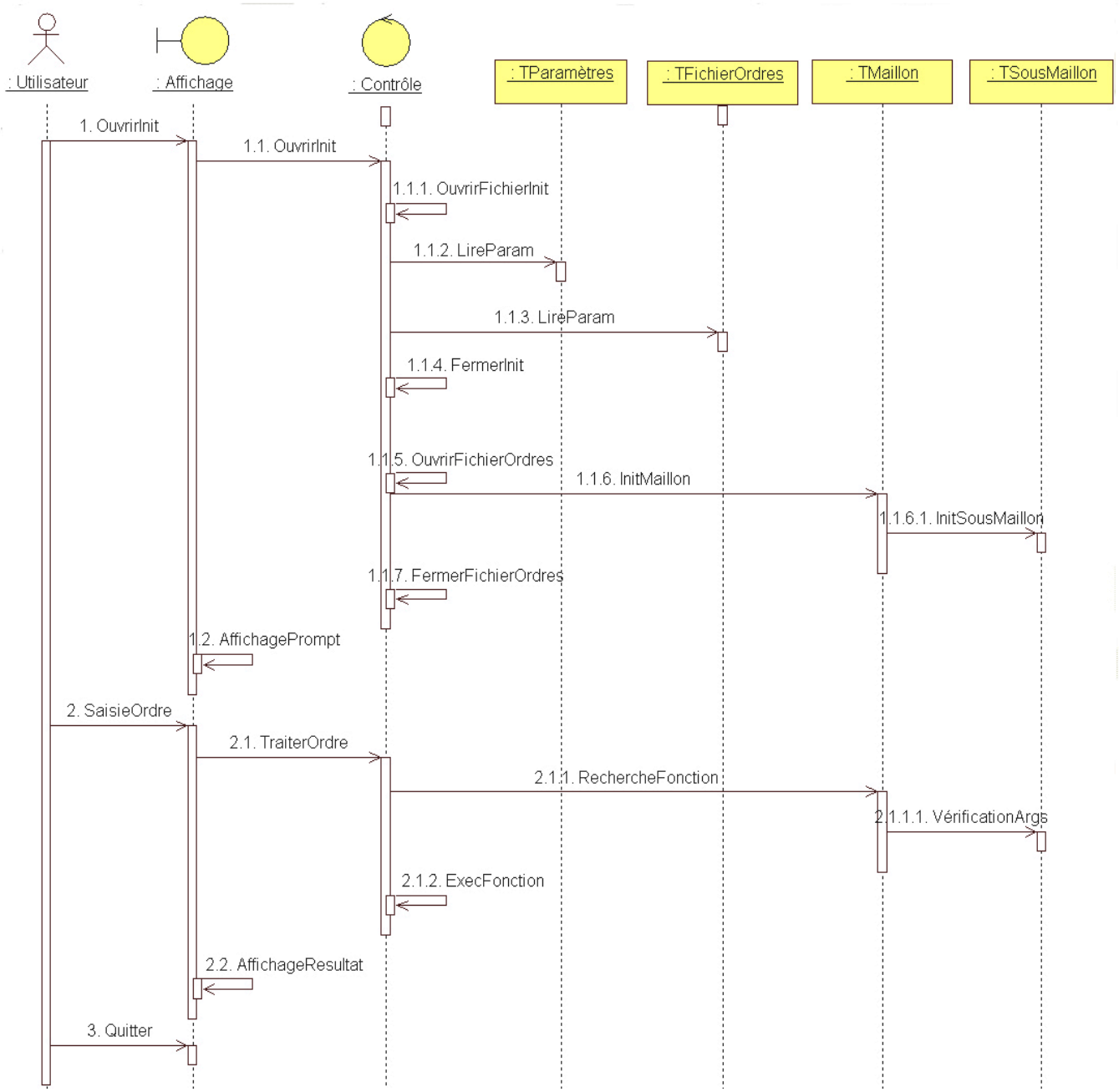
Un utilisateur (humain).
L'instrument lui-même.

3°/ Diagramme d'activités :

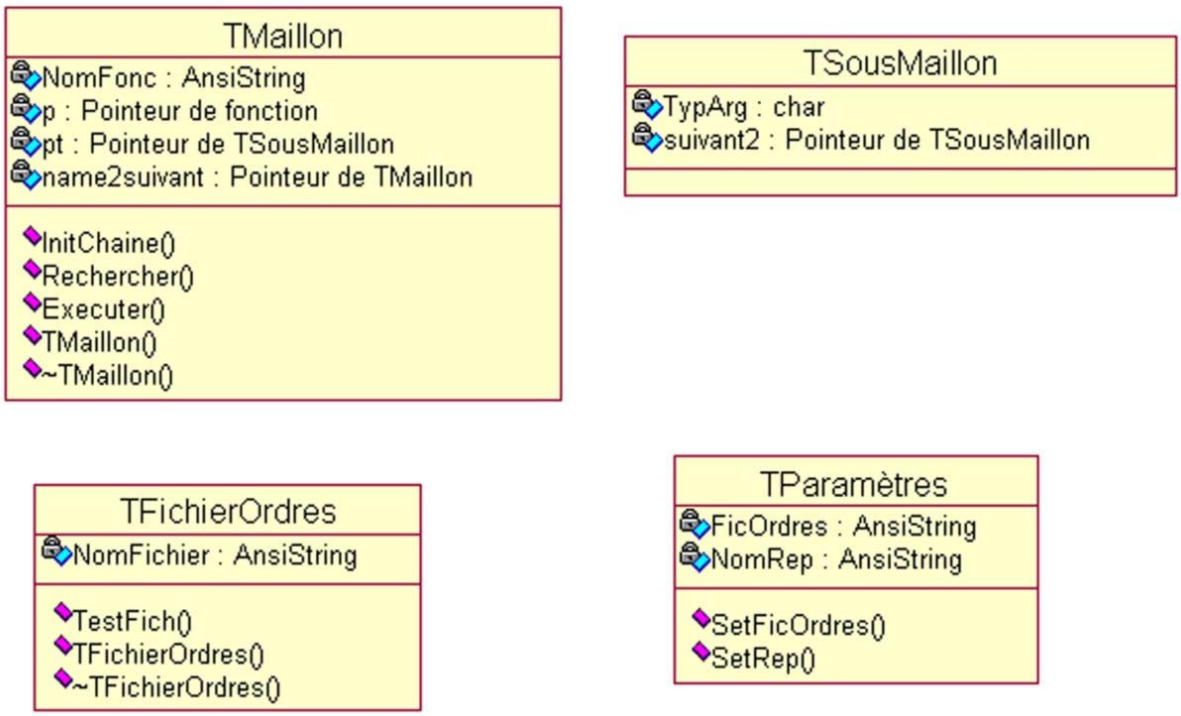
(Voir page suivante)


Ouverture de l'application
Ouverture du fichier d'initialisation
Initialisation du programme avec les paramètres du fichier d'initialisation
[Erreur initialisation avec le fichier d'ordres]
[Chargement des paramètres OK]
Ouverture du fichier d'ordres
Initialisation du programme avec le fichier d'ordres
[Erreur chargement paramètres]
[Initialisation OK]
Attente d'ordre (affichage d'un prompt)
[L'utilisateur ne quitte pas]
[Arrivée de l'ordre "QUIT"]
[Réception d'un ordre]
Vérification de l'existence de la fonction demandée
Demande de confirmation pour quitter le programme
Message d'erreur
[Fonction non existante]
[Fonction existante]
Vérification de la liste d'arguments
On passe à la fonction suivante pour les tests de validité
[Erreur dans la chaîne d'arguments]
[Arguments OK]
[Confirmation OK]
Le programme passe en paramètres la fonction et la liste d'arguments au synthétiseur
L'ordre a été exécuté
Fermeture de l'application

4°/ Diagramme de séquence :



5°/ Diagramme de classe :



Gestion des communication par protocole TCP

Le protocole TCP :

Les caractéristiques du protocole TCP :

TCP (qui signifie Transmission Control Protocol, soit en français: Protocole de Contrôle de Transmission) est un des principaux protocoles de la couche transport du modèle TCP/IP. Il permet, au niveau des applications, de gérer les données en provenance (ou à destination) de la couche inférieure du modèle (c'est-à-dire le protocole IP). Lorsque les données sont fournies au protocole IP, celui-ci les encapsule dans des datagrammes IP, en fixant le champ protocole à 6 (Pour savoir que le protocole en amont est TCP...). TCP est un protocole orienté connexion, c'est-à-dire qu'il permet à deux machines qui communiquent de contrôler l'état de la transmission.
Les caractéristiques principales du protocole TCP sont les suivantes:

- TCP permet de remettre en ordre les datagrammes en provenance du protocole IP,
- TCP permet de vérifier le flot de données afin d'éviter une saturation du réseau,
- TCP permet de formater les données en segments de longueur variable afin de les "remettre" au protocole IP,
- TCP permet de multiplexer les données, c'est-à-dire de faire circuler simultanément des informations provenant de sources (applications par exemple) distinctes sur une même ligne,
- TCP permet enfin l'initialisation et la fin d'une communication de manière courtoise.

Le but de TCP :

Grâce au protocole TCP, les applications peuvent communiquer de façon sûre (grâce au système d'accusés de réception du protocole TCP), indépendamment des couches inférieures. Cela signifie que les routeurs (qui travaillent dans la couche Internet) ont pour seul rôle l'acheminement des données sous forme de datagrammes, sans se préoccuper du contrôle des données, car celui-ci est réalisé par la couche transport (plus particulièrement par le protocole TCP).

Lors d'une communication à travers le protocole TCP, les deux machines doivent établir une connexion. La machine émettrice (celle qui demande la connexion) est appelée client, tandis que la machine réceptrice est appelée serveur. On dit qu'on est alors dans un environnement Client-Serveur.
Les machines dans un tel environnement communiquent en full duplex, c'est-à-dire que la communication se fait dans les deux sens.

Pour permettre le bon déroulement de la communication et de tous les contrôles qui l'accompagnent, les données sont encapsulées, c'est-à-dire qu'on ajoute aux paquets de données un en-tête qui va permettre de synchroniser les transmissions et d'assurer leur réception.

Description du module :

Ce module permet la communication des différentes machines entres elles. De plus, il permet l'accès aux différentes machines pour un utilisateur distant pour des raisons de diagnostic par exemple en cas de panne ou bien de prise de contrôle de l'instrument. Ce module permet d'abord l'établissement de la connexion entre les diverses machines puis de définir leur statut, à savoir, si elles sont machine cliente ou serveur. Pour MISOLFA, ces machines sont le PC de pilotage, le PC d'acquisition, le PC jouant le rôle de centralisateur et la console (qui peut ne pas être une machine). Le principe est relativement simple puisque la transmission de données se fera par l'intermédiaire de sockets. L'avantage est que le choix des machines en tant que client ou serveur ne se fera que d'un point de vue physique et non logique. En effet, l'ensemble du système doit rester opérationnel même si certaines machines ne fonctionnent pas. La transmission des données restant symétrique entre machines, le client « parle » au serveur et le serveur « parle » au client, d'un point de vue logique cela n'a pas d'importance. Ce module permet de détecter également les différents problèmes liés à une connexion distante comme l'établissement de la connection, savoir si le serveur est disponible, coupure intempèstive de la liaison ...

Un système de mots de passe sera également déployé afin de permettre la mise en place de priorités sur les connexions et de ne pas provoquer de problèmes liés à la connexion de plusieurs utilisateurs.

Analyse :

Borland C++ Builder possède quelques composants concernant les sockets. L'avantage de l'utilisation de sockets, c'est que, comme cité précédemment, d'un point de vue logique, n'ayant pas de différence entre la partie client et serveur, la mise en oeuvre de ce module en est plus aisé. En effet, l'échange de données sera exactement symétrique quelque soit le statut des machines.

Contrairement au niveau logique, la détermination des statuts de machines d'un point de vue physique est important. En effet, une partie du système doit pouvoir continuer à fonctionner quand certaines machines ne sont pas en état de remplir leur rôle. La fait que toutes les machines soient en relation impose donc une analyse logique du problème.

Les utilisateurs distants, qui utilisent donc une console (écran + clavier + souris) pour se connecter, sont obligatoirement clients. En se connectant aux machines d'acquisition et de pilotage, ces dernières jouent le rôle de serveur. Cette console est partagée, par l'intermédiaire d'un distributeur d'entrées / sorties, entre les différents PCs à des fins de diagnostic et d'intervention en cas de pannes ou de problèmes. On peut

également faire une prise de contrôle de l'instrument par ce biais là.
(Voir en Annexe 3 l'architecture informatique et électronique de SODISM II et MISOLFA)

Maintenant, d'un point de vue interne, il faut établir ces mêmes statuts (client-serveur) dans les communications. Le diagramme en étoile ci dessous résume les différents choix qui ont été pris. Ces choix ont été déterminé de par l'utilisation que l'on voulait donner à chaque machine.

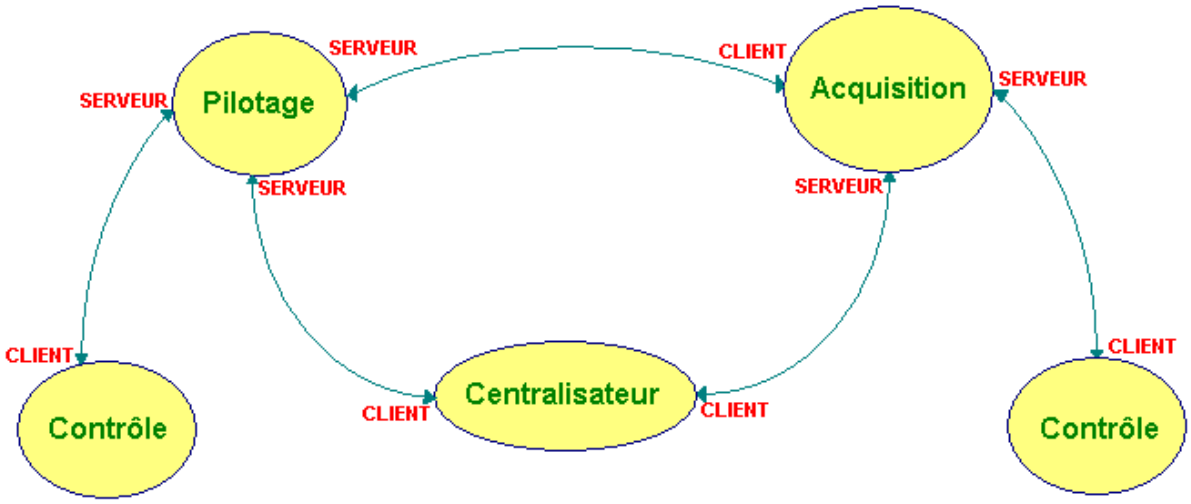


Schéma des statuts de connexion des machines composant MISOLFA

En effet, on peut voir par exemple que le statut du centralisateur est d'être client vis à vis du PC de pilotage et celui d'acquisition. Le rôle du centraliseur est de stocker toutes les informations essentielles à l'observateur pour le contrôle de l'ensemble des instruments. Ce centralisateur reçoit également des informations provenant d'autres instruments (DORAYSOL, SODISM II). L'observateur pourra également envoyer des commandes aux différentes machines. Le centralisateur a été choisi comme client car, dans le cas contraire, en cas de non-fonctionnement de la machine, tout le système ne pouvait fonctionner. Alors qu'en tant que client, la machine de pilotage et la machine d'acquisition resteront accessibles et pourront fonctionner correctment bien que l'observateur soit en mode « aveugle » (pas de retour et d'affichage d'informations concernant les différentes machines).

Il ne reste plus qu'un statut à déterminer entre le PC de pilotage et le PC d'acquisition. Une seule contrainte a permis de mettre en évidence le fait que le rôle du serveur soit tenu par le PC de pilotage. En effet, l'avantage de cette décision est que pour la ré-utilisation du téléscope dans un autre programme par exemple, cela ne poserait aucune difficulté. On sera donc capable de faire « pointer » l'instrument sans faire d'acquisition et donc de l'utiliser de manière indépendante face au PC d'acquisition.

Un système de mots de passe permet également une gestion plus fluide des connexions en cas d'utilisation du système par plusieurs utilisateurs. Cela évitera les conflits et donnera priorité à un seul utilisateur. De plus, cela rendra le système beaucoup plus sécurisé.

Les différentes interfaces graphiques

Description :

Ce module n'ayant pas encore été traité avant l'édition de ce rapport, aucune analyse ne sera fournie dans ce dernier. Cependant, une brève description du module sera donnée.

Ce projet, met en avant la création d'interfaces homme-machine nécessaire à aux différents modules. Ceci comprends l'interface graphique du centralisateur qui doit donc renvoyer toutes les informations utiles à l'observateurs. Ces informations proviennent des différentes machines composant le système PICARDSOL, c'est à dire, l'ensemble MISOLFA et d'autres instruments comme DORAYSOL ou bien encore SODISM II. Cette interface locale permettra également, en plus de l'affichage de données nécessaires, d'envoyer des commandes aux différentes machines. La réalisation de cette interface dans un langage précis est encore à décider. Cela se fera soit en C++ soit en JAVA.

La création d'une seconde interface graphique est également nécessaire puisqu'elle jouera le rôle d'interface déportée. En effet, celle ci est nécessaire lors d'une connexion distante. C'est pour cela que cette interface sera réalisée en JAVA grâce à ses qualités de portabilité et d'adaptabilité sur n'importe quel système. Cette interface sert à des fins de diagnostics en cas de problèmes au sein du système. Elle sert également à l'envoi de commandes aux différentes machines du système.

Conclusion

Ce stage m'a beaucoup apporté, aussi bien sur le plan de la vie professionnelle que dans la gestion de mon travail. Il m'a permis également de comprendre que la formation à l'IUT, bien qu'elle soit riche et variée, n'est en fait que des bases pour faciliter la formation constante à laquelle les différents métiers de l'informatique doivent faire face. En effet, j'ai pu constater que je devais accentuer mes compétences en matière de programmation pour pouvoir réaliser mon projet correctement. De plus, la recherche de documentation s'est avéré indispensable vis à vis des différents outils utilisés et de la compréhension générale du sujet. J'ai également pris conscience qu'une organisation correcte de mon travail était indispensable.

Mon sujet, m'a poussé à apprendre de nombreuses choses afin de maîtriser correctement tous les outils dont j'avais besoin. L'utilisation de structures de programmation complexes ou d'éléments que je ne connaissais pas m'a également poussé à parfaire les connaissances que je possédais déjà. Mais tous ces facteurs m'ont tout de fois ralenti dans la conception de l'Analyseur-Synthétiseur. En effet, une mauvaise organisation au départ m'a rapidement mis en difficulté. J'ai donc compris qu'une compréhension claire et exacte du sujet était nécessaire. Mon maître de stage aidant, j'ai pu donc reprendre une organisation correcte qui m'a permis de beaucoup mieux avancer qu'au début et également de prendre conscience de ce réel besoin.

Cette prise de conscience m'aidera sûrement dans le futur de ma profession et cette expérience a été pour moi plus qu'enrichissante. Je pense me sentir beaucoup plus « préparé » à la vie active bien que mon souhait soit de continuer encore mes études. Je pense que le bilan de mon expérience réalisée par le biais de ce stage me servira énromément durant ma poursuite d'études et dans ma carrière future.

Bibliographie et références

Livres et documentations :

Borland C++ Builder 5 pour Windows 2000 / 98 / 95 /NT
Le Guide du développeur
aux éditions Borland Inprise

Le Grand Livre du C++
de Werner ACHTERT
aux éditions Micro Application

MISOLFA : Un moniteur de qualité d'images solaires utile pour la mission PICARD
Abdanour IRBAH, Pierre ASSUS, Amokrane BERDJA, Julien BORGNINO, Maamar FODIL, Frédéric MORAND, Yacine SAIDI
CNES / CNRS - Service d'Aéronomie / OCA

Spécifications de la Mission PICARD
Gérard THUILLIER avec la collaboration de Bernard GELLY, Michel HERSE, Thierry APPOURCHAUX, Mireille MEISSONNIER
CNES / CNRS - Service d'Aéronomie / OCA

Automatique – Asservissement 1
de Philippe Bourzac
cours du lycée Hoche à Versailles (2001)

Sites Internet :

Le site de l'Observatoire de la Côte d'Azur :
www.obs-azur.fr

Le site du CNES :
www.cnes.fr

Le site du Laboratoire Universitaire d'Astrophysique de Nice :
www-astro.unice.fr

Le site de l'Université de Sophia-Antipolis :
www.unice.fr

Le site de l'astrorama (gestion des visites pour l'OCA)
www.astrorama.net

Le club d'entraide des développeurs francophones :
www.developpez.com

Répertoire des illustrations et photos

Liste des Acronymes

CERGA : **C**entre d'**E**tudes et de **R**echerches en **G**éodynamique et **A**stronomie

CNES : **C**entre **N**ational d'**E**tudes **S**patiales

CNRS : **C**entre **N**ational de la **R**echerche **S**cientifiques

DORAYSOL : **D**éfinition et **O**bservation du **RAY**on **SOL**aire

HRA : **H**aute **R**ésolution **A**ngulaire

IP : **I**nternet **P**rotocol

MISOLFA : **M**oniteur d'**I**mages **SOL**aires **F**ranco-**A**lgérien

OCA : **O**bservatoire de la **C**ôte d'**A**zur

SODISM I et II : **SO**lar **DI**ameter and **S**urface **M**apper

TCP : **T**ransmission **C**ontrol **P**rotocol

UMR : **U**nité **M**ixte de **R**echerche

Lexique

Astrométrie : Partie de l'astronomie qui étudie la position des astres.

DORAYSOL : Instrument d'observation permettant le mesure du diamètre solaire.

Haute Résolution Angulaire : Regroupe les techniques qui ont pour but de s'affranchir des effets de la turbulence atmosphérique sur l'observation céleste.

MISOLFA : Instrument permettant la mesure des paramètres de la turbulence atmosphérique.

Seconde d'arc : Unité de mesure d'angle. En astronomie, les angles apparents séparant les astres sont bien souvent inférieurs à un degré, il est donc nécessaire d'utiliser une mesure plus précise. Ainsi, un degré se décompose en 60 minutes d'arcs qui se décomposent elles-mêmes en 60 secondes d'arc.

SODISM I et II : Téléscope destinés à réaliser des images du diamètre solaire et des cartes de la surface du soleil.

ANNEXES

Annexe 1 :

Informations relatives à L'Observatoire de la Côte d'Azur

Adresses des sites :

Observatoire de la Côte d'Azur Calern	Observatoire de la Côte d'Azur Grasse	Observatoire de la Côte d'Azur Nice, Siège Social
2130, Route de l'Observatoire CAUSSOLS F-06460 St. VALLIER de THIEY Tél. : **+33 (0)4 93 40 54 54** Fax : **+33 (0)4 93 40 54 33**	Avenue Nicolas Copernic F-06130 GRASSE Tél. : **+33 (0)4 93 40 53 53** Fax : **+33 (0)4 93 40 53 33**	Boulevard de l'Observatoire B.P. 4229 F-06304 NICE Cedex 4 Tél. : **+33 (0)4 92 00 30 11** Fax : **+33 (0)4 92 00 30 33**

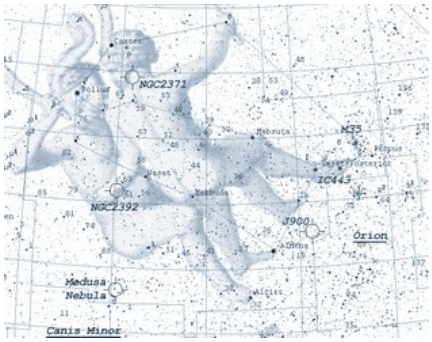
Coordonnées géographiques :

Site	Calern	Grasse	Nice
Longitude	6°56'00" Est	6°54'53.8" Est	7°18'1" Est
Latitude	43°45'00" Nord	43°39'54.9" Nord	43°43'11" Nord
Altitude	1270m (Schmidt)	532m (Accueil)	364m (Sommet)

Coordonnées du Laboratoire GEMINI :

Laboratoire GEMINI
Observatoire de la Côte d'Azur
Avenue Nicolas Copernic
06130 GRASSE - FRANCE

Tel: +33 (0)4 93 40 53 xx
Fax: +33 (0)4 93 40 53 33



Coordonnées de MORAND Frédéric (Site de Calern):

Frédéric MORAND
Téléphone : 04 93 40 54 36
e-mail : frederic.morand@obs-azur.fr

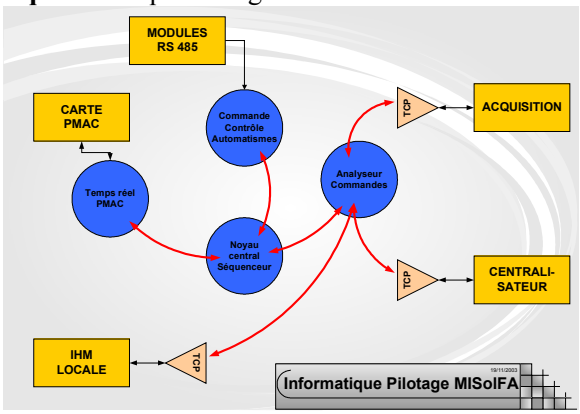
Annexe 2 :

Proposition de stage

Sujet : Conception et programmation de l'Interface Homme-Machine (IHM) et des communications internes et externes de l'instrument MISOLFA (mesure de la turbulence atmosphérique de jour)

Contexte : La mesure précise de la forme du Soleil, de ses dimensions et de leurs variations est un sujet d'étude important depuis environ 30 ans car elles sont encore très mal comprises. Jusqu'à présent, la quasi-totalité des mesures ont été faites au sol, où la turbulence atmosphérique limite notablement la précision. Le satellite PICARD, dont le lancement est programmé en 2008, doit se consacrer à l'étude du Soleil, et l'un de ses instruments, SODISM, a pour objectif la mesure du diamètre solaire hors atmosphère.
Pour bien comprendre l'influence de l'atmosphère sur la détermination du diamètre, il est prévu d'installer au sol, à l'Observatoire de la Côte d'Azur, une copie conforme de l'instrument SODISM et de faire observer simultanément les deux instruments durant toute la mission du satellite. Un moniteur de turbulence atmosphérique de jour, MISOLFA, sera adjoint à SODISM 2 et permettra de modéliser quantitativement les effets de l'atmosphère sur la détermination du diamètre. MISOLFA doit procéder à ses premiers tests opérationnels au cours du deuxième semestre 2004 et entrer en fonctionnement scientifique début 2005. C'est le développement de l'IHM de MISOLFA et de ses communications internes et externes qui fait l'objet de la présente proposition de stage.

Problématique : D'un point de vue informatique, MISOLFA est constitué de deux PC, l'un dit d'**acquisition**, qui contrôle les systèmes d'enregistrement des données (caméra et photodiodes), l'autre dit de **pilotage**, qui contrôle les moteurs de l'instrument (mouvements du télescope, roues à filtres, …) au travers d'un **rack électronique de pilotage**. L'interface de l'ordinateur de pilotage avec le rack se fait par l'intermédiaire d'une carte électronique (PMAC) qui gère l'asservissement des moteurs et d'une liaison série RS485 qui sert à la commande et au contrôle des automatismes.
Le logiciel de l'ordinateur de pilotage est centré autour d'un **noyau séquenceur** qui interagit avec :
- un **module temps réel** qui contrôle la carte PMAC,
- un **module RS485** pour la liaison série qui gère les automatismes,
- un « **analyseur-synthétiseur** » qui permet l'interface avec les autres parties du système. Cet analyseur-synthétiseur analyse les messages qu'il reçoit ou génère des commandes qu'il envoie. Ces messages ou commandes sont sous forme de « phrases » ASCII, constituées de mots-clés et de paramètres. Ces messages ou commandes transitent par des liens éthernet (protocole TCP-IP), gérés sur chaque machine par l'intermédiaire de sockets.



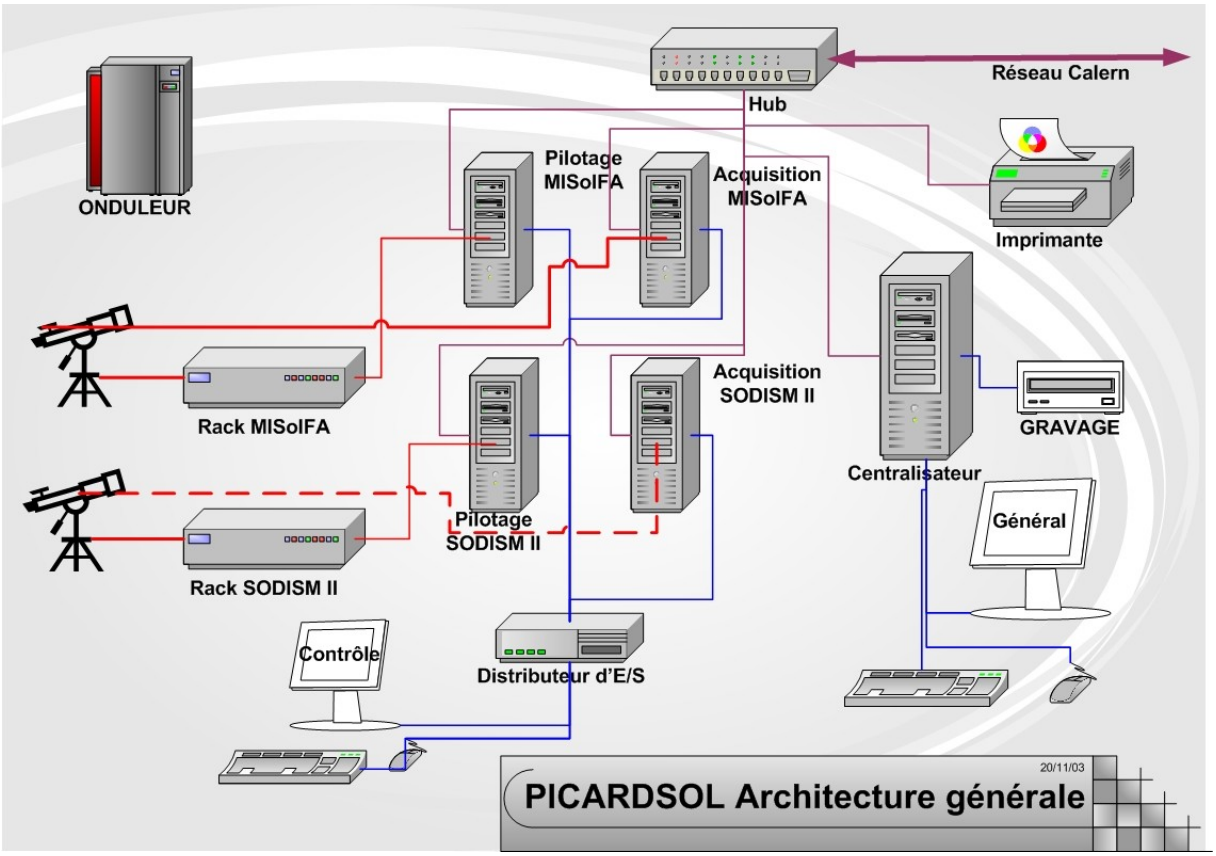
Ces liens doivent être établis avec :
- l'ordinateur d'acquisition,
- une **IHM locale**, qui peut se situer a priori sur n'importe quelle machine,
- un **centralisateur**, ordinateur qui regroupe toutes les informations importantes provenant de MISOLFA, SODISM 2 et DORAYSOL (l'actuel instrument de mesure du diamètre).

Travail demandé au stagiaire : Le stagiaire devra réaliser :
- l'analyseur-synthétiseur de commandes, ainsi que ses adaptations sur le centralisateur, le PC d'acquisition et pour l'IHM locale,
- l'IHM locale,
- la mise en œuvre et la gestion des liens entre les différentes machines (connexions, sécurité, conflits de priorité, …).

Ces éléments seront réalisés en C++ ou en JAVA.
Les modules temps réel (PMAC, RS485) et le séquenceur ne seront pas réalisés par le stagiaire.

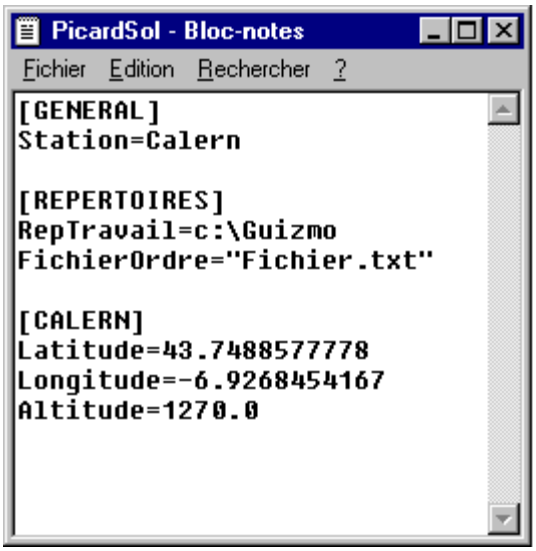
Annexe 3 :

Architecture informatique et électronique générale de PICARDSOL (SODISM II + MISOLFA)



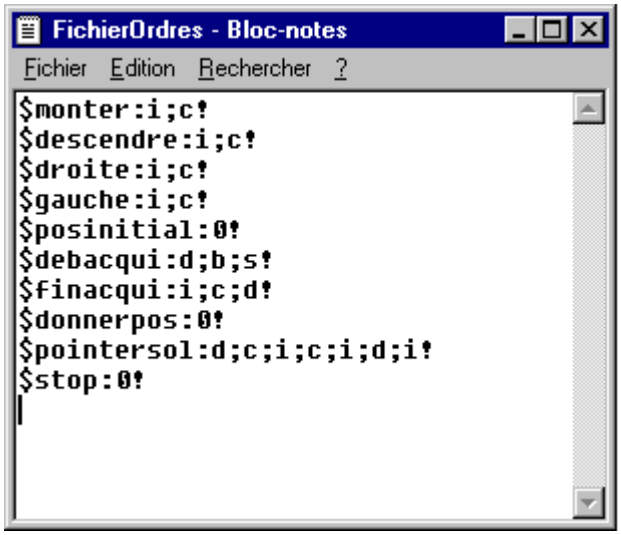
Annexe 4 :

Exemple d'un fichier d'initialisation pour l'Analyseur-Synthétiseur



Annexe 5 :

Exemple d'un fichier d'ordres pour l'Analyseur-Synthétiseur



Annexe 6 :

Diagramme de l'initialisation de l'analyseur avec le fichier .ini